Patent **Attorney Docket No.: 2207/9800**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANTS	:	Stephan J. JOURDAN et al.
SERIAL NO.	:	09/708,722
FILED	:	November 9, 2000
FOR	:	INSTRUCTION SEGMENT RECORDING SCHEME
GROUP ART UNIT	:	2183
EXAMINER	:	Aimee J. LI

M/S: APPEAL BRIEF – PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

APPEAL BRIEF

Dear Sir:

This brief is in furtherance of the Notice of Appeal, filed in this case on November 18, 2009.

1. REAL PARTY IN INTEREST

The real party in interest in this matter is Intel Corporation. (Recorded September 16, 2004; Reel/Frame 015789 / 0667).

2. RELATED APPEALS AND INTERFERENCES

Appeal Briefs were previously filed in this case on September 7, 2005 and December 16, 2006.

3. STATUS OF THE CLAIMS

Claims 1-19 are pending, rejected and on appeal. No claims are withdrawn, objected to, or allowed. No amendments to the claims were made after the Final Office Action dated August 18, 2009.

The claims in their current form (including those claims under appeal) are presented in The Appendix – Section 8 – Claims on Appeal.

4. STATUS OF AMENDMENTS

The claims listed on page A-1 of the Appendix attached to this Appeal Brief reflects the present status of the claims.

5. SUMMARY OF THE CLAIMED SUBJECT MATTER

The present invention relates to a recording scheme for instruction segments in a processor core in which instructions from instruction segments may be cached in reverse

program order.

The embodiment of independent claim 1 generally describes a cache comprising: a cache line to store an instruction segment further (*e.g.*, *see* page 4, lines 5-6 – Figure 2, 210) comprising a plurality of instructions stored in sequential positions of cache line (*e.g.*, *see* page 4, lines 5-6 – Figure 2, 210) in reverse program order (*e.g.*, *see* page 3, line 25 to page 4, line 2 – Figure 2), wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream (*e.g.*, *see* page 5, lines 9-10 – Figure 4, IP_2 and IP_4) causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line (*e.g.*, *see* page 5, lines 24-25 – Figure 4).

The embodiment of independent claim 5 generally describes a segment cache for a front-end system in a processor (*e.g.*, *see* page 4, lines 5-6 – Figure 2, 210), comprising a plurality of cache entries to store instructions of instruction segments in reverse program order (*e.g.*, *see* page 3, line 25 to page 4, line 2 – Figure 2), wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream (*e.g.*, *see* page 5, lines 9-10 – Figure 4, IP_2 and IP_4) causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line (*e.g.*, *see* page 5, lines 24-25 – Figure 4).

The embodiment of independent claim 8 generally describes a method comprising: building an instruction segment based on program flow (*e.g.*, *see* page 4, lines 5-6 – Figure 2, 210), and storing instructions of the instruction segment in a cache entry in reverse program order (*e.g.*, *see* page 3, line 25 to page 4, line 2 – Figure 2), wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a

second instruction stream (*e.g.*, *see* page 5, lines 9-10 – Figure 4, IP_2 and IP_4) causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line (*e.g.*, *see* page 5, lines 24-25 – Figure 4).

The embodiment of independent claim 14 generally describes a processing engine, comprising: a front end stage to build and store instruction segments (*e.g.*, *see* page 1, line 7 – Figure 1, 110), instructions provided therein in reverse program order (*e.g.*, *see* page 3, line 25 to page 4, line 2 – Figure 2), wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream (*e.g.*, *see* page 5, lines 9-10 – Figure 4, IP_2 and IP_4) causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line (*e.g.*, *see* page 5, lines 24-25 – Figure 4), and an execution unit in communication with the front end stage (*e.g.*, *see* page 4, line 10).

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Are claims 1-5, 8 and 11-14, rejected under 35 U.S.C. § 102(b), anticipated by Peleg et al., (hereinafter "Peleg"), US Pat. No. 5,381,533?

B. Are claims 6-7 and 15-19, rejected under 35 U.S.C. § 103(a), unpatentable over Peleg, US Pat No. 5,381,533, in view of Rotenberg et al., (hereinafter "Rotenberg"), "A Trace Cache Microarchitecture and Evaluation", IEEE © 1999?

C. Are claims 9-10, rejected under 35 U.S.C. § 103(a), unpatentable over Peleg, US Pat. No. 5,381,533, in view of Kyker et al., (hereinafter "Kyker"), US Pat. No. 6,578,138?

7. ARGUMENT

A. Claims 1-5, 8 and 11-14 rejected under 35 U.S.C. § 102(b) are not anticipated by Peleg.

Applicants submit the cited references do not teach, suggest or disclose at least "[a] cache comprising: a cache line to store an instruction segment further comprising a plurality of instructions stored in sequential positions of the cache line in reverse program order, wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line" (*e.g.*, as described in claim 1).

The Examiner asserts Peleg teaches the relevant limitations at column 1, line 64 to column 2, line 19, column 4, lines 11-23, column 8, line 30 – column 9, line 35, column 10, lines 1-24, and column 10, line 58 to column 11, line 12, and cites Figure 5A, 5B, and 6-12 generally. *See* Office Action dated 8/18/2009, paragraph 11. Applicants disagree.

The first cited section, column 1, line 64 to column 2, line 19, consists of the entire Summary of the Invention section. It generally describes that the cited reference is directed to storing data in a cache memory. More specifically, it states it is directed to identifying trace segments of instructions in a computer program in the order they are executed. Applicants submit it does not describe the specific, relevant limitations of claim 1.

The second cited section, column 4, lines 11-24, is similarly general. It describes that a "basic block" as defined by the reference comprises instructions in a computer program which are unconditionally and consecutively executed and begins after execution of branch instruction

and ends with the execution of another branch instruction. It further states that once a first instruction in a basic block is executed, all the remaining instruction in the basic block will for certain be executed since there are no instructions in the basic block that go beyond an inserted branch instruction. Applicants submit this cited section does not teach or suggest at least the relevant, specific limitations of claim 1 discussed above. For example, contrary to the Office Action's assertion, this section fails to teach or suggest at least "…wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line" (*e.g.*, as recited in claim 1). *See* Office Action dated 8/18/2009, paragraph 7.

Next, the Examiner cites to the extensive section of column 8, line 30 – column 9, line 35. Applicants submit that because of the extensive nature of the section, Applicants have no choice but to discuss the section in summary. . Applicants maintain the cited section fails to teach or suggest the relevant limitations. It is directed to the description of Figures 5A, 5B. Figure 5A describes a static sequence of computer program instructions and the flow that may occur during execution of the instructions. FIG. 5B illustrates the order of execution of blocks of the instruction for the flow of FIG. 5A. The cited section sequentially describes in detail execution by the computer of program instruction according to the method described (*e.g.*, "…At the end of this block the branch is taken and BB.sub.N+1 is next executed (BB.sub.3). At the end of BB.sub.3 the branch is not taken and BB.sub.4 is executed…"). *See e.g.*, column 8, lines 30-43. Applicants submit that similar to the cited sections discussed above, this cited section fails to teach or suggest at least a conditional branch causing program flow to jump from a first location

in a first instruction stream to a second location in a second instruction stream causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line" (*e.g.*, as described in claim 1).

The Examiner also cites to column 10, lines 1-24 which fail to teach or suggest the relevant limitations as well. This cited section describes Figure 9, which, similar to the previous cited section describes the sequential execution of computer program instructions according to the described method. It describes, for example, storing blocks twice in the array. The reference states this assures that the CPU can access a block in cache even where the branching is different than predicted. It also describes specific steps as described in Figures 9 and 10 (*e.g.*, "As shown in FIG. 9, after BB.sub.4 is executed it is transferred into the line 75 in a position after BB.sub.3 and into the beginning of line 73. As shown in FIG. 10, BB.sub.3 and BB.sub.4 are transferred to the array along with the address A.sub.3 and the next address A.sub.12. After execution of BB.sub.4, the CPU will request the instruction at A.sub.12. When it does, a hit will occur as shown in FIG. 10."). Again, Applicants submit this cited section fails to teach or suggest the relevant limitations of claim 1 discussed above.

The last cited section, column 10, line 58 to column 11, line 12 generally describes the benefits from implementing the methods as described in the reference. In sum, it states that if multiple blocks are repeatedly executed (in a loop), the instructions in the loop will be continuously supplied to the CPU with only a single address being sent to the cache memory. However, this section fails to teach or suggest the specific limitations of claim 1 discussed above.

B. Claims 6-7 and 15-19 rejected under 35 U.S.C. § 103(a) are not unpatentable over Peleg, US Pat No. 5,381,533, in view of Rotenberg, "A Trace Cache Microarchitecture and Evaluation", IEEE © 1999.

C. Claims 9-10 rejected under 35 U.S.C. § 103(a) are not unpatentable over Peleg, US Pat. No. 5,381,533, in view of Kyker, US Pat. No. 6,578,138.

Rotenberg and Kyker fail to make up for the deficiencies of Peleg. Rotenberg is directed to increasing fetch bandwidth and decreasing performance bottlenecks. Kyker is directed to various methods of storage and unrolling of instruction loops. Applicants submit these references fail to make up for the deficiencies of Peleg for at least the reasons discussed in previous response, and hereby maintain and incorporate by reference all arguments made in previous responses.

Therefore, since for at least the above-discussed reasons, the cited references fail to teach or suggest each and every limitation of claim 1, they fail to support proper § 102 or § 103 rejections. Applicants submit claim 1 is allowable. Independent claims 5, 6, 8 and 14 contain similar allowable limitations, and are therefore allowable for similar reasons. Claims 2-4, 7, 9-13 and 15-19 are allowable for depending from allowable base claims.

Appellants therefore respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's decision rejecting claims 1-19 and direct the Examiner to pass the case to issue.

The Examiner is hereby authorized to charge any additional fees which may be necessary for consideration of this paper to Kenyon & Kenyon Deposit Account No. **11-0600**.

Respectfully submitted,

KENYON & KENYON LLP

Date: January 19, 2010

By: /Sumit Bhattacharya/
Sumit Bhattacharya
(Reg. No. 51,469)

KENYON & KENYON LLP
333 West San Carlos St., Suite 600
San Jose, CA 95110

Telephone: (408) 975-7500
Facsimile: (408) 975-7501

APPENDIX

(Brief of Appellants Stephan J. JOURDAN et al.
U.S. Patent Application Serial No. 09/708,722)

8. CLAIMS ON APPEAL

The claims in their current form (including those claims under appeal) are presented below:

1. (Previously Presented) A cache comprising:

a cache line to store an instruction segment further comprising a plurality of instructions stored in sequential positions of cache line in reverse program order, wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line.

2. (Previously Presented) The cache of claim 1, wherein the instruction segment is an extended block.

3. (Previously Presented) The cache of claim 1, wherein the instruction segment is a trace.

4. (Previously Presented) The cache of claim 1, wherein the instruction segment is a basic block.

5. (Previously Presented) A segment cache for a front-end system in a processor, comprising a plurality of cache entries to store instructions of instruction segments in reverse

program order, wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream causes a terminal instruction from the second instruction stream to be stored in a first position of a cache line.

6. (Previously Presented) The segment cache of claim 5, further comprising:

an instruction cache system,

an instruction segment system, comprising:

a fill unit provided in communication with the instruction cache system,

and

a selector coupled to an output of the instruction cache system and to an output of the segment cache.

7. (Previously Presented) The segment cache of claim 6, wherein the instruction segment system further comprises a segment predictor provided in communication with the segment cache.

8. (Previously Presented) A method comprising:

building an instruction segment based on program flow, and

storing instructions of the instruction segment in a cache entry in reverse program order, wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream causes a terminal

instruction from the second instruction stream to be stored in a first position of a cache line.

9. (Original) The method of claim 8, further comprising:

building a second instruction segment based on program flow, and

if the first and second instruction segments overlap, extending the first instruction segment to include non-overlapping instructions from the second instruction segment.

10. (Original) The method of claim 9, wherein the extending comprises storing the non-overlapping instructions in the cache in reverse program order in successive cache positions adjacent to the instructions from the first instruction segment.

11. (Original) The method of claim 8, wherein the instruction segment is an extended block.

12. (Original) The method of claim 8, wherein the instruction segment is a trace.

13. (Original) The method of claim 8, wherein the instruction segment is a basic block.

14. (Previously Presented) A processing engine, comprising:

a front end stage to build and store instruction segments, instructions provided therein in reverse program order, wherein a conditional branch causing program flow to jump from a first location in a first instruction stream to a second location in a second instruction stream causes a

terminal instruction from the second instruction stream to be stored in a first position of a cache line, and

an execution unit in communication with the front end stage.

15. (Previously Presented) The processing engine of claim 14, wherein the front-end stage comprises:

an instruction cache system,

an instruction segment system, comprising:

a fill unit provided in communication with the instruction cache system,

a segment cache, and

a selector coupled to an output of the instruction cache system and to an output of the segment cache.

16. (Previously Presented) The processing engine of claim 15, wherein the instruction segments are extended blocks.

17. (Previously Presented) The processing engine of claim 15, wherein the instruction segments are traces.

18. (Previously Presented) The processing engine of claim 15, wherein the instruction segments are basic blocks.

19. (Previously Presented) The processing engine of claim 15, wherein the instruction segment cache system further comprises a segment predictor provided in communication with the segment cache.

9. EVIDENCE APPENDIX

No further evidence has been submitted with this Appeal Brief.

10. RELATED PROCEEDINGS APPENDIX

Per Section 2 above, there are no related proceedings to the present Appeal.